System Services CSCI

Network Services CSC

Redstone

Design Panel 3 Review

June 19, 1997

1. Network Services CSC

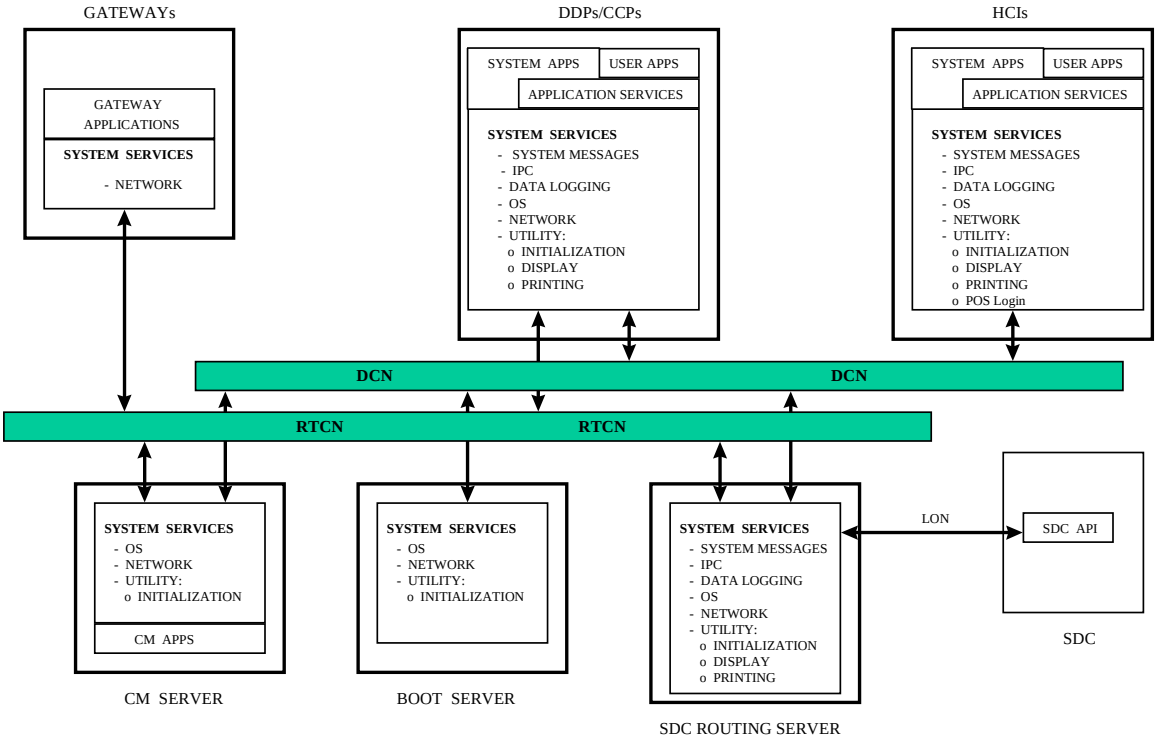
1.1 Network Services Introduction

1.1.1 Network Services Overview

The Network Services CSC for Redstone will include the following:

- Basic Communication Service
- Network APIs
- Activity Separation

These services are resident on the RTPS platforms as outlined below.



SYSTEM SERVICES OVERVIEW

1.1.2 Network Services Operational Description

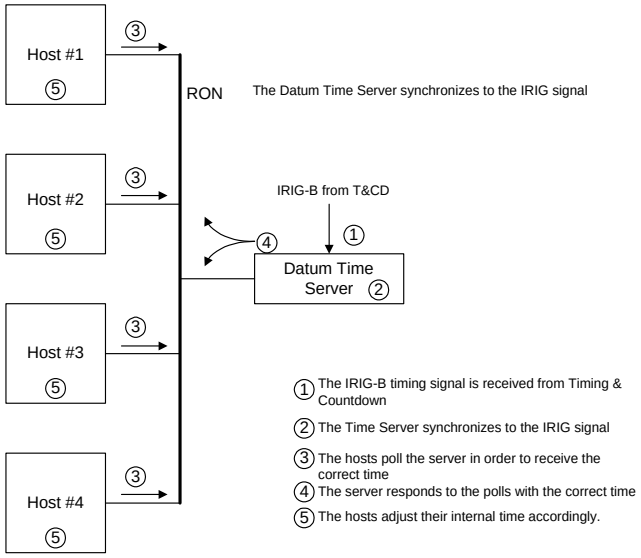
The Network Services provides the capability by which end system applications, services and users communicate and distribute data within CLCS. The three major elements of the Network Services are: Basic Communication Services, Network APIs, and Activities & Activity Separation.

1.1.2.1 Basic Communication Services:

The Basic Communications Services, listed below, consist of LAN services and protocols which are supplied by the COTS OS on each CLCS platform.

- Transmission Control Protocol/ Internet Protocol (TCP/IP): TCP/IP is provided as a COTS product.
- File Transfer Protocol (FTP): The FTP protocol is provided as a COTS product.

- Telephone Network (TELNET): Telnet is provided as a COTS product.
- Network Information Service (NIS): The NIS will be implemented by a NIS server platform that is resident in each Flow Zone. This server will have direct connectivity to each Operational network (RTCN, DCN) within its associated Flow Zone and any attached Control and Gateway Groups. It will provide password control for users utilizing the resources within the string of equipment.
- Network File System (NFS): The Network File System is provided as a COTS product. It's configuration is controlled by the Operating Systems Group.
- Network Timing Protocol (ntp): The Network Timing Protocol provides a mechanism to synchronize the clocks residing on local hosts with both a common timing source and each other. The common timing source will be a Datum timing box that accepts an IRIG-B timing signal from Timing and Countdown (T&CD) and functions as an ntp server. This Datum box will be present in each Flow Zone and act as the time reference for all local hosts contained within that Flow Zone as well as any Control Groups that may be attached to that Flow Zone.



Network Timing Protocol Flow
(Redstone only)

- UNIX r-commands (e.g., rlogin): The Unix r-commands are provided as a COTS product.

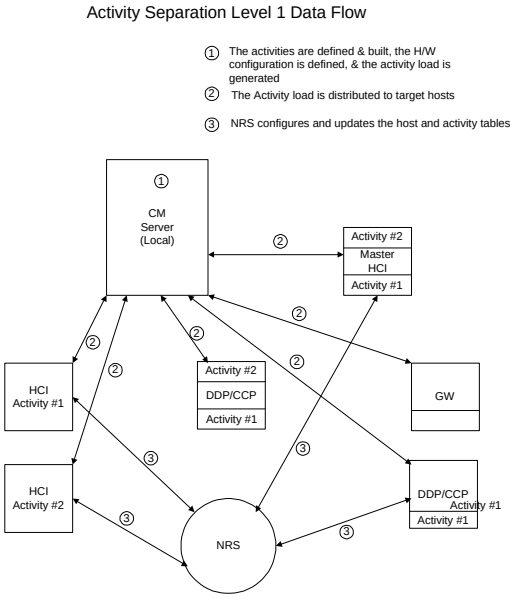
1.1.2.2 Network APIs:

The Network APIs consist of a common set of custom developed library calls which applications can use for data distribution.

Application Messaging consists of a set of APIs for Connection-Oriented Point-To-Point communications. Client Applications will be able to establish and open a connection with applications on a remote server machine. Once the connection is established, data can be sent/received to/from the remote machine.
Connectionless Messaging consists of a set of APIs for Connectionless Point-To-Point communications. Local applications will use CLM for sending datagrams Point-To-Point to a specific host machine - no confirmation for transmission or acknowledge of delivery is required from the receiver. The CLM API will also support the transmission of Reliable Multicast data - one transmitter to a group of receivers. For Redstone, the multicast data will be communicated at the UDP layer. **The Multicast CLM will be enhanced with the reliable functionality.**
Network Registration Service (NRS). A method for determining the IP Multicast addresses, assignment of file descriptors, the location of applications and the distribution of this information is required in order for applications to send and receive data on the network. For Redstone, static registration files or tables will be the mechanism used on the RTCN and for multicast address resolution . The registration files or tables will be maintained through manual procedures. The NRS will be utilized on the DCN for point-to-point communications. NRS allows applications to register their existence by hostname and port number. Other applications can then locate these applications through the NRS APIs. This information is broadcasted by each local NRS process onto the network whereby, the NRS process on the other machines update their local NRS information.

1.1.2.3 Activities & Activity Separation:
Activity will be a term utilized to classify an operation in the CLCS. A Naming Standard for data streams will be developed which uses parameters from the Activity definition (i.e., TCID, SCID, Vehicle or Tail Number, Live

Vehicle or Sim, etc.). Logical Separation can then be made by allowing applications in one Activity to only communicate on their Activity.



An activity will be defined via the OPS CM Activity Manager. When a set of hardware (Gateway Group(s), Control Group(s), Flow Zone(s)) is assigned to support a particular activity, the CM Server will download the activity load to the specific HWCI's supporting that activity utilizing an Activity Manager function. As a part of the load procedure, the host tables for each HWCI will be manipulated (utilizing NRS) in such a manner that only those HWCI's that constitute that particular activity will have visibility to only those HWCI's within their activity. HWCI's that provide services common across activities (ie: NIS servers, etc…) will maintain their capability to see (and be seen by) all other platforms.

In addition to supporting host file manipulations, NRS will also maintain the Static Multicast Table. This table defines the stream name to multicast address associations. Also, a mapping from stream name to Service ID's for any required point to point communications will be maintained. These stream names will be the only communications paths available within an activity.

Redstone assumption: There will be no redundancy implemented (particularly on the DCN) for Redstone. The DCN (and potentially the RTCN) implementation calls for two independent networks. NRS will need changes made to it in order to support this configuration. This is out of scope and will be implemented during Thor.

1.2 Network Services Specifications

1.2.1 Network Services Groundrules

- The Network Services API will be designed with the following RTPS LAN requirements into consideration:

    A) The RTPS LAN will be fault tolerant, such that, a single LAN component will not cause more than one end station failure.
    B) The switching of a LAN component to a backup (due to a failure) will be deterministic and within the Reliable Message time allocation.
    C) Fault isolation and LAN component replacement will be conducted with minimum interference to the on-going operation.
    D) An industry LAN standard protocol for supporting one-to-many communications will be utilized, such that, interoperability among different LAN products can be obtained.

- **Redundancy within the RTCN and DCN will not be implemented during Redstone. The issue will be studied and at the time of the Redstone delivery a recommendation will be made as to the optimal course of action. There are many factors which will influence the design of the operational networks redundancy scheme, including:**
    A) **SDC recording issues.**
    B) **NIS implementation.**
    C) **Failover implementation on the RTCN.**
    D) **Multiple networks comprising the DCN.**

1.2.2 Network Services Functional Requirements

NOTE 1: Requirements in this document in *Italics* format are Post Redstone and are specified for design consideration and future delivery capabilities (not to be tested in Redstone).

The Network Services Functional Requirements area is composed of the following CSC functions:

1. Network APIs
2. Network Registration Services API
3. Activity Separation (Data)
4. Activity Separation (Platform)
5. Basic Communications

1.2.2.1 Network APIs

1. Network Services will provide applications with a common API for communicating across the network interfaces.
2. The Network Services API will provide independent transmit and receive functionality.
3. The API will be capable of reliably sending multicast and broadcast data streams.
4. The API will be capable of supporting connection oriented point-to-point data, connectionless point-to-point data, and connectionless point-to-multipoint data.
5. The Network Services API will provide a mechanism to associate applications with Service Points (SP's).
6. The API will support the fragmentation and reassembly of messages that exceed the physical layer Maximum Transmission Unit (MTU).
7. Data messages will be re-transmitted to end stations registered for reliable message communication participation
8. Re-transmission of the current data message will complete or time-out prior to transmission of the next message received from the sending application
9. At a minimum, failed attempts and repeated retries will be reported to the calling application for transfer to System Messages.
10. The number of data message transmission retries will be statically configurable per data stream type. (Note: Entries will be procedurally controlled to prevent exceeding the system data transmission time limitations)
11. Message delivery attempts by the sender will be aborted based on a configurable expiration timer set for acknowledgment responses.
12. The API will provide automatic sequential numbering of data packets based on stream or port connections
13. The API will contain the capability to receive and buffer the next application message(s) until the current message is transmitted, re-transmitted or aborted.
14. The receiver will discard the whole message if an error is detected or message re-assembly is unsuccessful.
15. The receiver will report receive errors to the calling application for transfer to System Messages.
16. In cases where the source of the stream has moved from one platform to another or has closed a stream and then re-opened the same stream, the API will allow receiving applications to continue to receive a data stream without having to register again.
17. Reliable message for connection oriented (point-to-point) communication will be provided as specified by the TCP layer standards.
18. The sender will transmit the multicast and broadcast data even though a client has not yet registered to participate in reliable messaging communication.
19. The API will associate the stream name (defined in the Naming Standard) with a unique multicast address. The stream name assignment to multicast addresses will consist of a static file for Redstone.
20. The API will allow applications to open multiple streams within a single process.
21. The API will provide a function which relieves applications from having to poll for received data.
22. The API software will be structured and designed such that a single software baseline can be obtained regardless of platform type (i.e. VxWorks or UNIX).
23. ***The multicast acknowledgments will be transferred to SDC for recording.***
24. ***The API will contain the ability to enable/disable the transfer of acknowledgments to SDC for recording, with enable as the default setting.***

1.2.2.2 Network Registration Service API:

1. The Network Registration Service API will provide the following functions:

    a) A capability that allows applications to add an entry to the registration file.
    b) A locate function that allows applications to locate a registered file entry.
    c) A de-register function that allows applications to remove a registered entry.

2. Registration information will be distributed to all the pertinent platforms assigned to a specific Test Set.
3. All transactions which alter host tables will be logged.
4. Dynamic update of host tables will be provided based on the configured activity on each platform.
5. In point -to-point communications the Network Registration Service will dynamically allocate the port for a given application.
6. Registration of platform name only (without port name) will be permitted.
7. The association of platform and activities will be recorded.
8. The association of platform and activities will be provided through an API.
9. Deactivation of an activity will not interfere with activity resources in use.
10. The following API calls will be used to support registration functions:

    a) register
    b) get port
    c) port register
    d) de-register
    e) search
    f) port search
    g) platform list
    h) activate activity
    i) de-activate activity
    j) get activity
    k) get activity list

11. The Network Registration Service will log all API requests and major network events.
12. The Network Registration Service will provide the capability to register separate point-to-point service IDs by activity.

1.2.2.3 Activity Separation (Data)

1. A naming convention will be provided which maps or is associated to a unique multicast data stream.
2. Receivers will be able to only input the data streams associated with its selected activity.

1.2.2.4 Activity Separation (Platform):

1. Logical separation of platforms will be maintained on the network between different types of activities.
2. A capability will be provided to keep platforms supporting one activity type invisible to platforms supporting another activity.
3. The data separation function will allow certain positions to communicate with devices regardless of the activity selected. Note: These positions are said to be “Global”.
4. Separation of point-to-point data will utilize the platform separation.
5. The activity-independent or Global workstation and servers will be visible to all workstations at all times.
6. Uncommitted workstations will not have access to workstations and servers committed to an activity.
7. Each workstation will have access to all workstations and servers which are part of the same activity.

1.2.2.5 Basic Communication Services:

1. The Basic Communication Services will provide the functionality of the following services as defined by the COTS TCP/IP Standards:

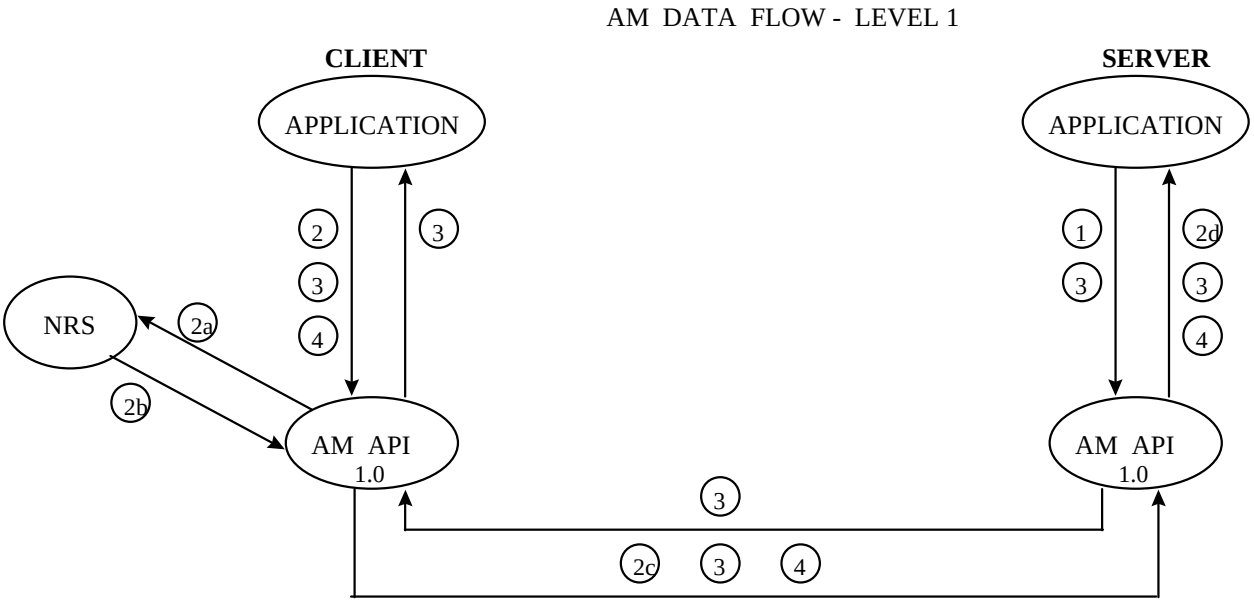
a) File Transfer Protocol (ftp)
b) Telnet
c) Unix r-commands
d) Network File System (NFS)
e) Network Timing Protocol (NTP)
f) Network Information Service (NIS)

2. A command line interface will be provided for the ftp, Telnet, and Unix r-command services.
3. *The Basic Communication Services function will support the Simple Network Management Protocol (SNMP).*
4. Network Services will provide logical communications such that applications on the same platform can communicate transparently as if they were remote on the network. This is in support of CLCS Application Groupings or Location Transparency onto a single platform

1.2.3 Network Services Performance Requirements

1. The Network Registration Service will allow remote de-registration of a specified service identifier (ID) in less than two seconds.
2. The Network Registration Service will Propagate new registrations in less than two seconds.

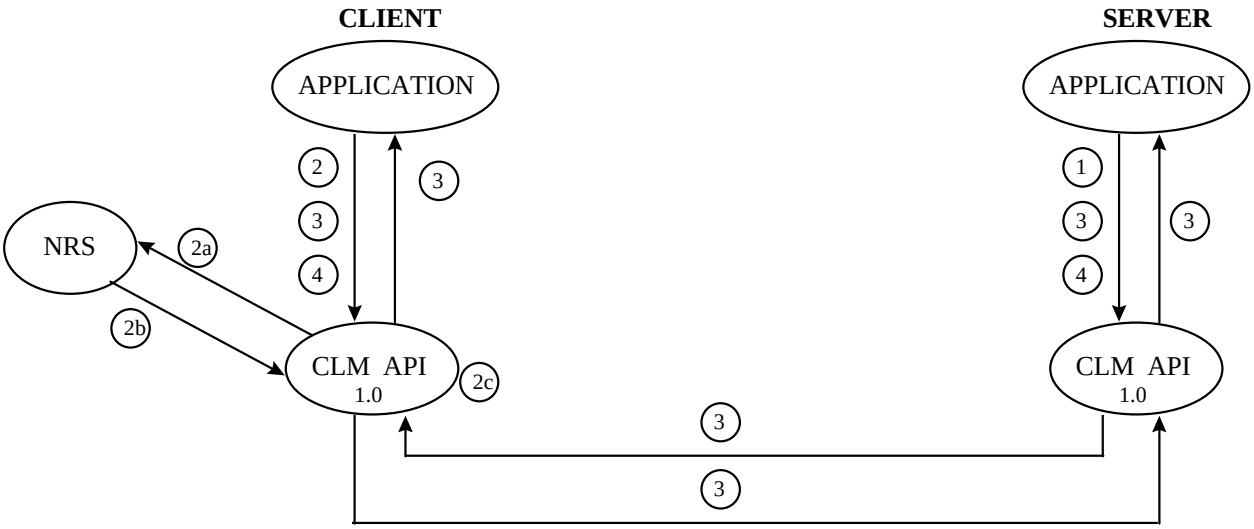
1.2.4 Network Services Interfaces Data Flow Diagrams



1. Server Application Opens an AM socket
2. Client Application Opens an AM socket :
   - 2a AM searches NRS for Server location if requested by application
   - 2b NRS returns information
   - 2c AM connects to server application via TCP
   - 2d Server accepts connection socket
3. Client or Server transmits or receives data
4. Client closes connection and AM notifies the server process

CLM DATA FLOW - LEVEL 1

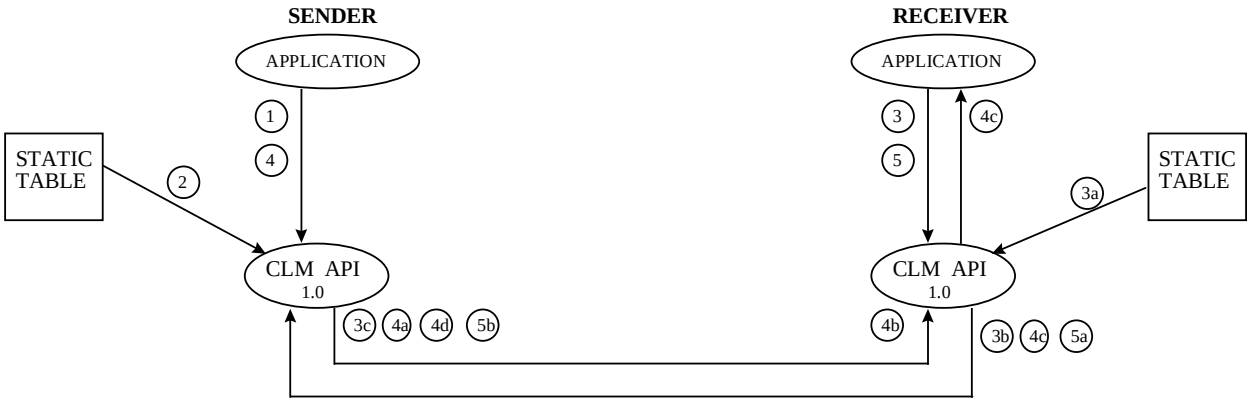
(UNRELIABLE POINT-TO-POINT & BROADCAST)



1. Server Application Opens a CLM P-P or Broadcast socket
2. Client Application Opens a CLM P-P or Broadcast socket :
   - 2a. CLM searches NRS for P-P Server location if requested by application
   - 2b. NRS returns P-P information
   - 2c. Client API sets up destination address
3. Client or Server transmits or receives data
4. Each appliation closes the CLM socket

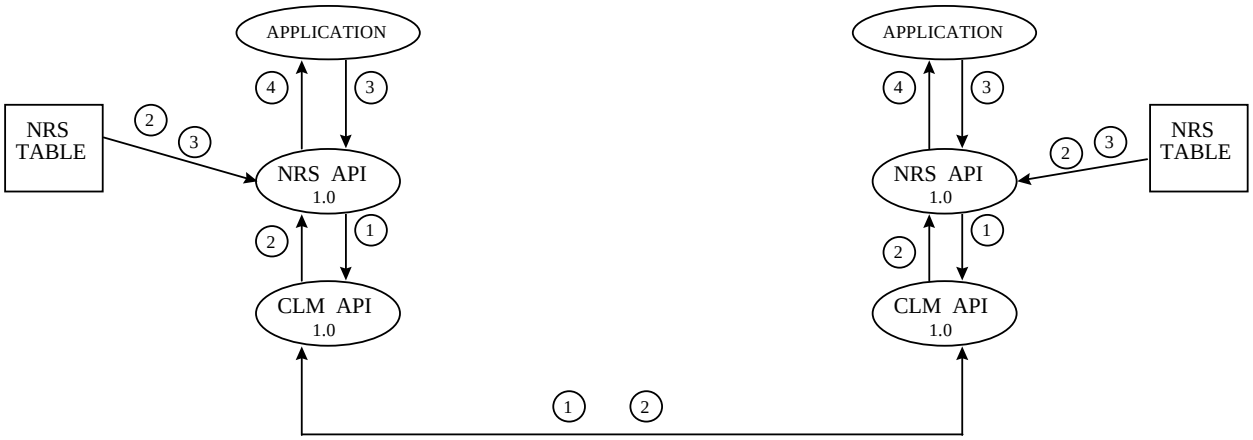
CLM DATA FLOW - LEVEL 1

(RELIABLE MULTICAST & BROADCAST)



1. Sender Opens a Reliable Multicast or Broadcast CLM socket
2. CLM reads the Static Table to map the Service Name to the IP Address
3. Receiver opens a reliable multicast CLM socket
   - 3a. CLM reads the Static Table to map the service name to the IP Address, Port Number and Source IP Address
   - 3b. Receiver sends a request to the Sender to be added to the list of receivers for the specific stream
   - 3c. Sender acknowledges the request
4. Sender transmits Multicast or Broadcast data
   - 4a. CLM calculates error-checking algorithim & sends the data over the specified IP Multicast socket or Broadcast Address
   - 4b. Receiver inputs and validates the packets, including verifying the error-checking algorithim.
   - 4c. Receiver sends a Point-To-Point ACK to the sender & passes the data to the application
   - 4d. Sender waits on ACKs from all Receivers. If one ACK missing, it re-sends the IP Multicast or Broadcast packet
5. Receiver closes its reliable Multicast CLM socket
   - 5a. Receiver sends a request to the sender to remove itself from the data stream list
   - 5b. Sender acknowledges the request

NRS DATA FLOW - LEVEL 1



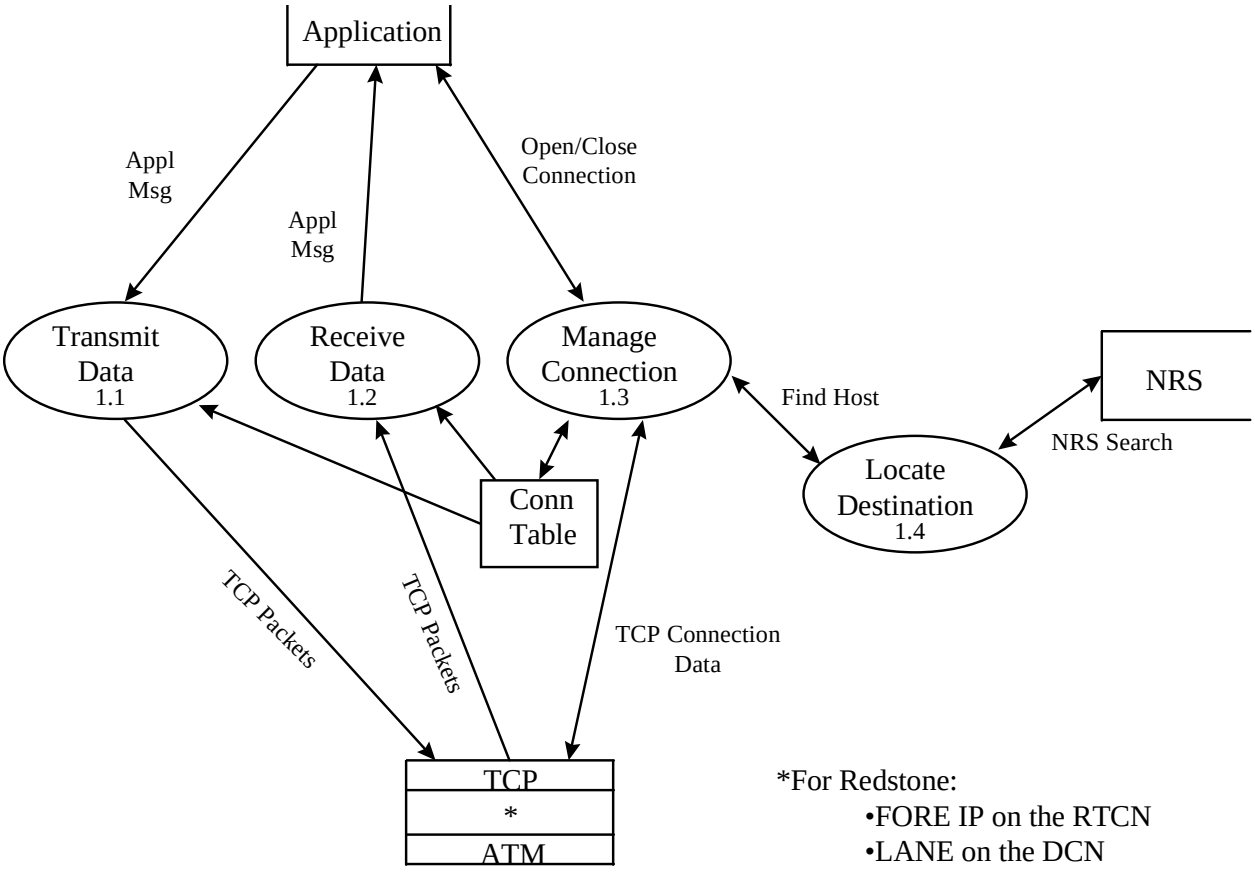
1. Each NRS Server broadcasts its registered Service IDs + Activity data via CLM
2. Each NRS Server receives all NRS broadcasts and updates its local memory NRS Table
3. Applications call NRS APIs to add, delete, or search the NRS data
4. NRS returns status or data

1.3 Network Services Design Specification

1.3.1 Network Services Detailed Data Flow

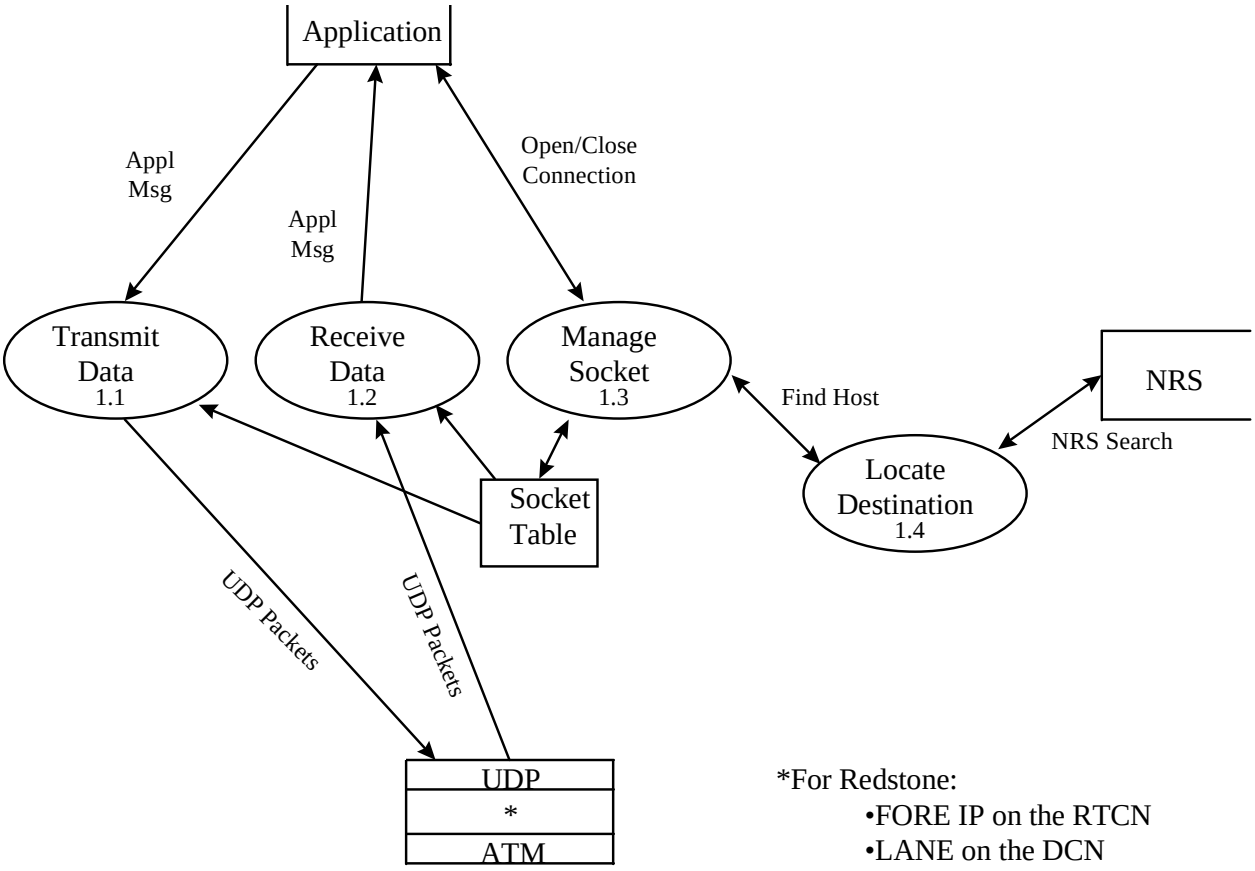
APPLICATION MESSAGING

DATA FLOW DIAGRAM - LEVEL 2



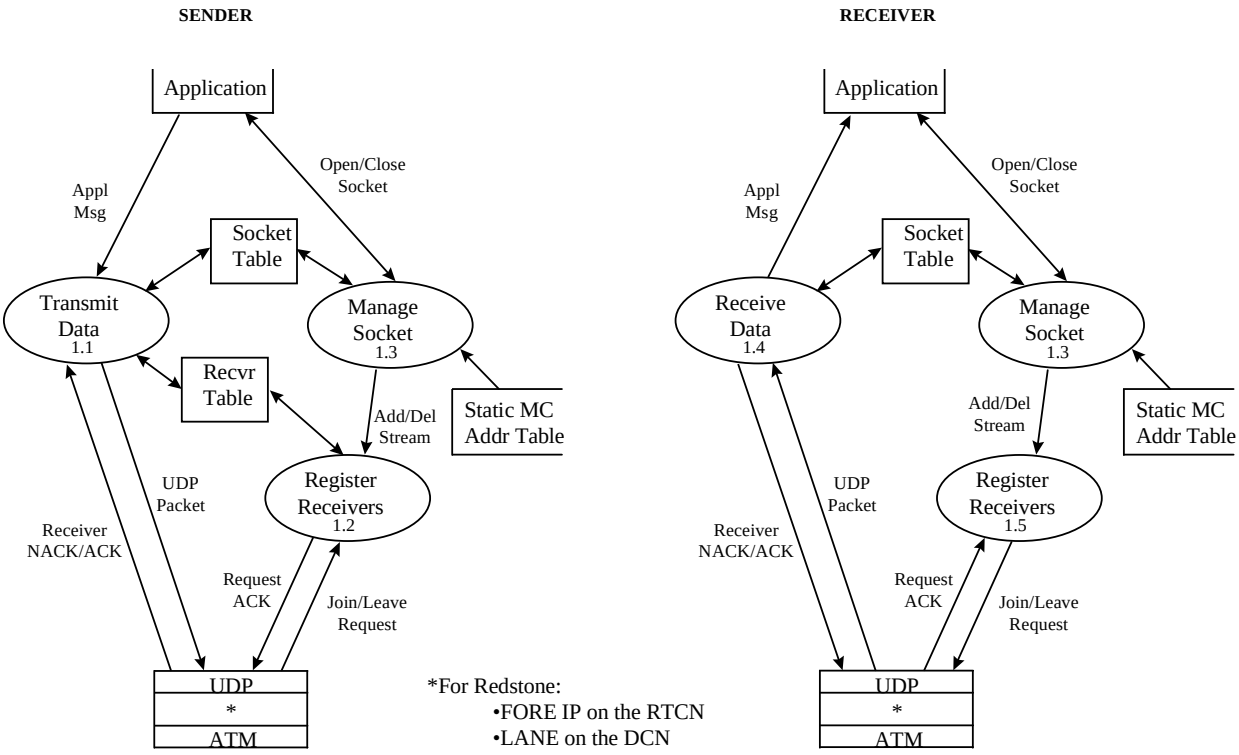
CONNECTIONLESS MESSAGING (Broadcast, Point-to-Point)

DATA FLOW DIAGRAM - LEVEL 2



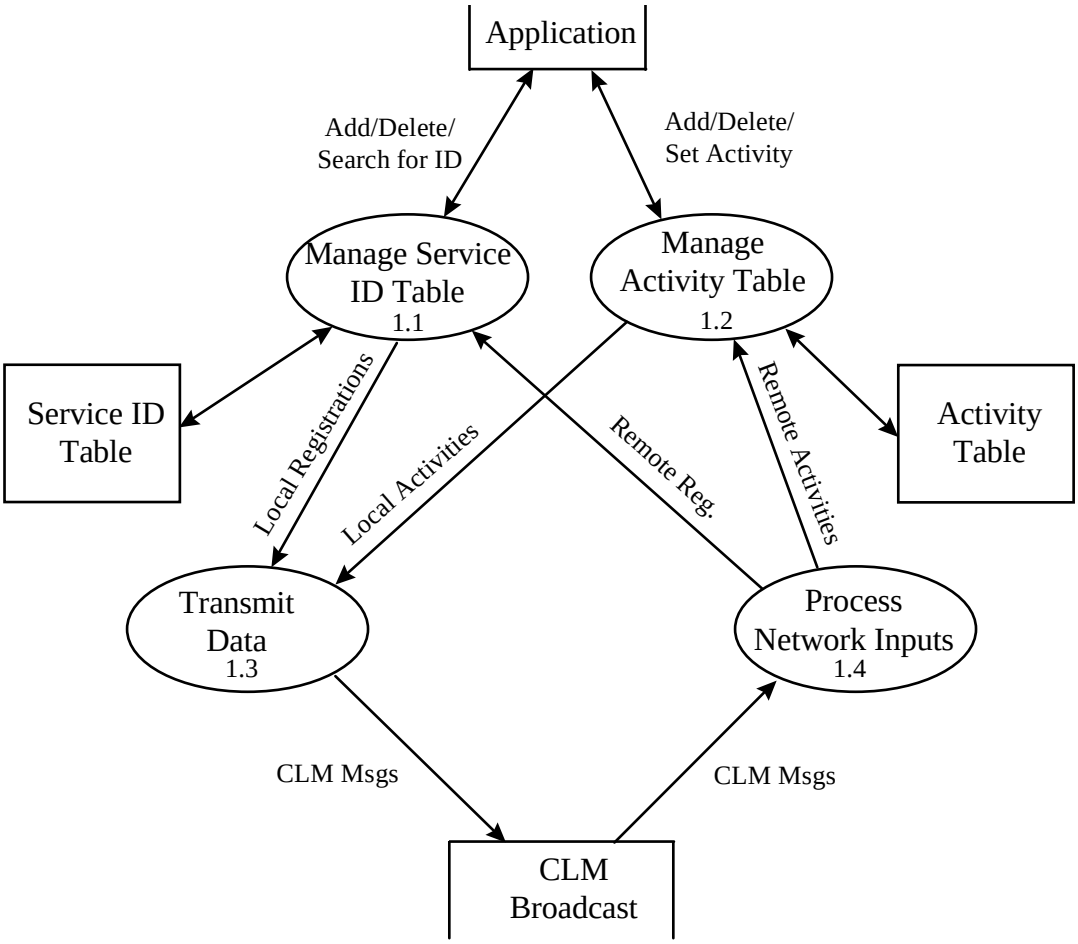
*For Redstone:
- •FORE IP on the RTCN
- •LANE on the DCN

CONNECTIONLESS MESSAGING (Reliable Multicast)
DATA FLOW DIAGRAM - LEVEL 2



*For Redstone:
- •FORE IP on the RTCN
- •LANE on the DCN

NETWORK REGISTRATION SERVICE
DATA FLOW DIAGRAM - LEVEL 2



1.3.2 Network Services External Interfaces

1.3.2.1 Network Services Message Formats

Since this CSC is strictly a set of library calls provided for the use of calling applications, no System Messages will be generated by the Network Services CSC. Rather, error codes will be generated and passed to the calling applications. The rationale for this implementation is that the overhead required to set up a System Message connection and to generate packets for transmission is beyond the scope of a library call.

NOTE: There have been changes to the error codes since the last DP

```
#define  RM_ERRBASE 2000              /* Set ridiculously high for now ..*/

#define  NOTSND       RM_ERRBASE+0    /* attempt to send on receive stream */
#define  NOTRCV       RM_ERRBASE+1    /* attempt to receive on send stream */
#define  TIMEOUT      RM_ERRBASE+2    /* no response from 1 or more dest within timeout period*/
#define REXMIT        RM_ERRBASE+3    /*Retransmission limit exceeded*/
#define  BADRSP       RM_ERRBASE+4    /* unknown response opcode from dest */
#define  MAXRCV       RM_ERRBASE+5    /* max no of receivers currently on */
```

1.3.2.2 Network Services Display Formats

This section is not applicable to the Network Services CSC

1.3.2.3 Network Services Input Formats

This section is not applicable to the Network Services CSC.

1.3.2.4 Recorded Data

This section is not applicable to the Network Services CSC.

1.3.2.5 Network Services Printer Formats

This section is not applicable to the Network Services CSC.

1.3.2.6 Interprocess Communications (C-to-C Communications?)

This section is not applicable to the Network Services CSC.

1.3.2.7 Network Services External Interface Calls (e.g., API Calling Formats)

The Network Services API is fully defined in the **Network Services Interface Definition Document (84K00354)**. Provided below is a complete list of the available calls with their parameters. For a full definition, refer to the IDD.

AM/CLM Calls:

- am_open(node, service, type)
- am_connect(sd)
- am_accept(sd)
- am_close(sd)
- am_recv(sd, bufptr, bufsize, log)
- am_send(sd, bufptr, nbytes, log)
- am_get_client(sd)
- am_set_timeout(t)

- clm_open(node, service, type)
- clm_close(sd)
- clm_recv(sd, bufptr, bufsize, log)
- clm_send(sd, msgptr, msgsize, log)
- clm_get_addr(sd)
- clm_set_addr(sd, sp)
- idd_set_bufsize(sd, type, rsize, ssize)
- idd_select(sd, sel, timeout)

NRS Calls:

- nrs_register(service_id)
- nrs_get_port(service_id, port_id)
- nrs_port_reg(service_id, port_id)
- nrs_deregister(service_id)
- nrs_server_deregister(service_id)
- nrs_search(service_id, host_ids)
- nrs_port_srch(service_id, host_ids, port_ids)
- nrs_node_list(node_ids, activity_ids)
- nrs_activate_act(activity)
- nrs_deactivate_act(activity_id)
- nrs_get_act(index, activity)
- nrs_get_act_table(activity)
- nrs_ws_config_act(activity_id)
- NRS_aix_gethostname() **(obsolete)**
- nrs_remote_port_reg(service_id, host_id, act, port_id)
- nrs_remote_cmd(service_id, host_id, cmd_type, act)
- nrs_service_name_srch(service_name, host_ids, serv_ids, port_ids)
- nrs_act_type_node_srch(option, host_ids, act_types, activity_ids)

1.3.2.8 Network Services Table Formats

The only table generated from a source external to the Network Services CSC is the Static Address Table. Each communications path within the system is named in this file and a mapping from this name to the multicast address/port numbers for data and acknowledgments associated with this path is provided. The proposed format for each entry is as follows:

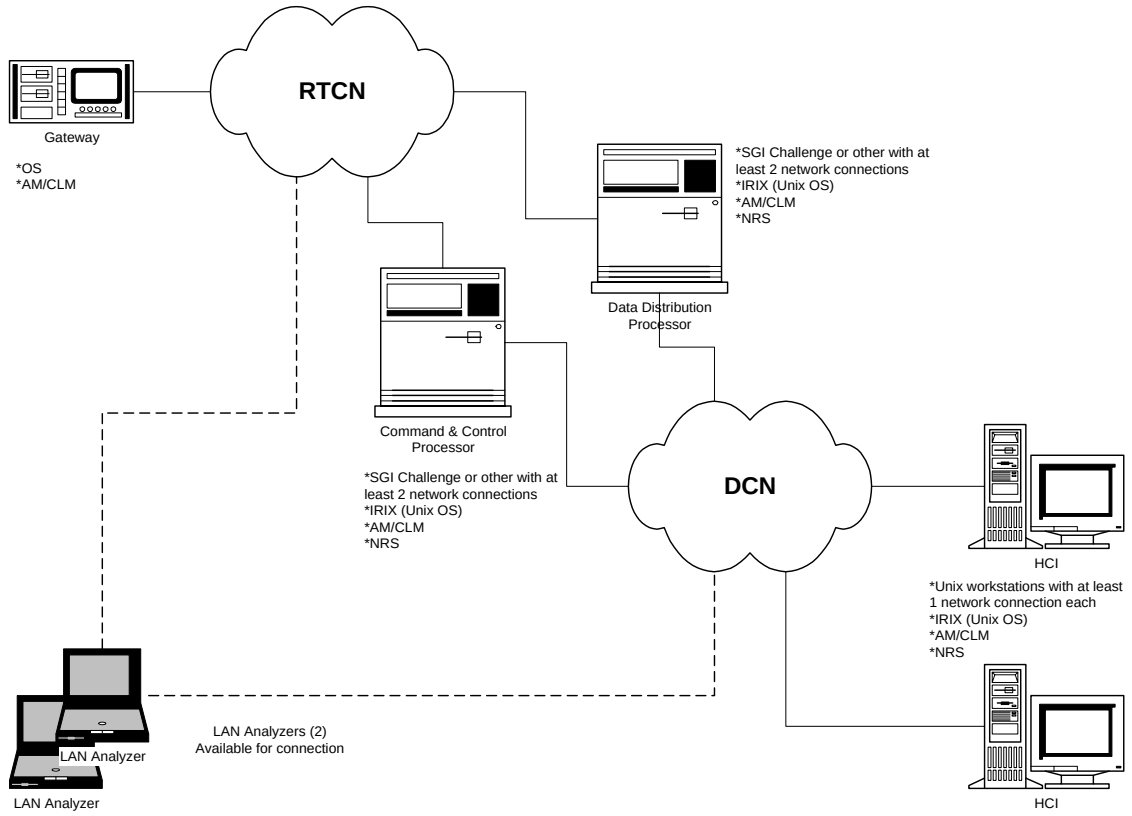
<stream_name> <data_ip_address> <data_udp_port> <ack_ip_address> <ack_udp_port>

where:
stream_name is the name of the data stream (this format is TBD),
data_ip_address is the address of the data stream (broadcast or multicast),
data_udp_port is the UDP port number associated with the data stream,
ack_ip_address is the address to which ack's are sent by the receiver,
and **ack_udp_port** is the UDP port to which ack's are sent by the receiver.

This table is currently generated by hand. It must be present on all hosts that utilize the Reliable Messages protocol. This static address file format allows us to define, at initialization time, whether ack's will be point-to-point or multicast, on a per stream basis. This should prove useful when deciding whether or not to record each stream, and for performance testing.

1.3.3 Network Services Test Plan

A detailed test procedure will be developed that will address the particular network services requirements of the Redstone delivery. The Hardware and software requirements are displayed in the following diagram.



1.3.3.1 Test Environment

The #2 Shuttle Development Environment (SDE2) facility will be used to test the network services. All processor platforms and network switches that will be required are available as part of the SDE. Each system involved shall be loaded with a baseline image of the Operating System, the network API's, and any test scripts to be executed. In addition, the network elements (i.e. switches) will be loaded with a TBD configuration prior to testing.

The SDE equipment actually participating in the test shall have all network connections removed or disabled except those that are part of the test. This is to verify that the communication is taking place over the CLCS physical networks and not over the currently installed Ethernet.

1.3.3.2 Test Tools

In addition to the working environment of equipment where network services reside, up to two LAN analyzers may also be required for some test procedures. These analyzers will need the ability to capture individual network frames or cells as well as to re-assemble these units into packets from higher level COTS protocols.

1.3.3.3 Test Cases

1.3.3.3.1 Basic Communication Services: The Basic Communication Services are provided by COTS software which will be subject to a Qualification Test (QT) to verify that they meet CLCS platform requirements. This will be achieved through command line activation or verification of the following components:

a) File Transfer Protocol (FTP)

b) Remote Login (Telnet)
c) Unix r-commands
d) Network File System (NFS)
e) Network Timing Protocol (NTP) ??
f) Network Information Service (NIS) ??

1.3.3.3.2 Network APIs: Network API testing will be accomplished via the development of simple code sequences that will exercise the AM/CLM calls and NRS calls. These procedures may be run with a network analyzer attached to verify proper retry and timeout thresholds are being upheld, and that packet formats and framing are as expected.

1.3.3.3.3 Activities & Activity Separation: The functionality of activity separation will be tested at an application level through the use of code sequences to simulate application calls. COTS tools (ping) will be utilized to verify that platforms within a common activity can communicate with each other and that platforms that are in separate activities can not communicate. An extended verification may be conducted at the physical level with the aid of a network analyzer.